Embedded control and virtual instrument simplifies laboratory automation

J. Jayapandian

Personal computers (PCs) are playing a vital role everywhere. From physics to psychology, the use of PCs is increasing rapidly. At present the trend in laboratory as well as industrial automation, to establish ongoing research and development programmes is minimal hardware design and maximal support of software. The software availability and the intelligent embedded hardware design configuration enable easy interfacing of systems with PCs. In the field of research and development programmes PCs act as a powerful tool for measurement of data and automation of experiments through their standard interfacing techniques. Virtual instrument design concept in software provides graphical user interface and enables a simple design solution with embedded controller through its standard communication interface with PCs. This note describes automation of some simple physics experiments using PCs.

The advancement in solid state technology to function in all possible environmental conditions provides a clear pathway for innovation in research and development. The programmable embedded design plays a major role in modern technological development. Visualization and implementation of required interface design with the intelligent programmable embedded controls like FPGA, ASIC, DSP, PSoC, microcontrollers, etc. provides an easy means of user-friendly design automation. On the other hand, it is possible to design simple laboratory/industrial automation according to our requirements by exploiting personal computers (PCs) with the availability of innovative software and programmable hardware components. Generally, for any laboratory as well as industrial automation, the tasks required for automation are sensing and controlling physical variables like position, temperature, pressure, flow, etc. Most of the variables are analogue in nature, with varying signal amplitude. Control over the basic variable parameter is an essential step for experimental automation. This note describes how the embedded control and virtual instrument software simplifies experimental automation using PCs through its standard interfacing technique.

Automation

Automation is an interfacing technique which provides interaction between a PC or any intelligent device and laboratory equipment, to get reliability, accuracy and remote operation. PCs are digital machines; they can accept only 'zeros' or 'ones'. Zero represents zero state in binary system (i.e. 0 volt state) and one represents one state (i.e. 5 volt state). But the physical variables are in analogue form. Hence the first step of interfacing requirement is analogue to digital (binary) conversion of the physical quantity to be monitored. Once the variables become digital information, it is easy to interact with a PC for further connectivity of the system which contains various physical quantities/variables to be monitored and measured. Various interfacing techniques can interact with the PC for experimental automation. Some of them are serial interface, parallel interface, Universal Serial Bus (USB) interface, game port interface, etc. The next section provides detailed specifications about each interfacing standard.

Interfacing standards

PCs serial interface

Serial ports are a type of computer interface that comply with the RS-232 standard. They are nine-pin connectors that relay information, incoming or outgoing, one byte at a time. Each byte is broken up into a series of eight bits, hence the term serial port. Serial ports are one of the oldest types of interface standards. Before internal modems became commonplace, external modems were connected to computers via serial ports, also known as communication or 'COM' ports. Computer mice and keyboards also use serial ports. Some serial ports use 25-pin connectors, but the nine-pin variety is more common. Serial ports are controlled by a special chip called a UART (Universal Asynchronous Receiver Transmitter). Serial ports differ from 25-pin parallel ports in that the parallel ports transmit one byte at a time using eight parallel wires; each carrying one bit. With data travelling in parallel, the transfer rate is greater. A parallel port could support rates up to 100 kilobytes per second, while serial ports only support 115 kilobits per second (kbps). Later, enhanced technology pushed serial port speeds to 460 kbps. This interface standard permits remote system handling and permits an easy way of interfacing with PCs.

Parallel port interface

LPT (Line Printing Terminal) is the common name given to a parallel port on IBM and compatible machines. Although the DB-25 female connector at the back of a PC is referred to as an LPT port, technically an LPT port is simply a parallel port set to LPT(x) with an I/O (input/output) address and IRQ (Interrupt Request) assigned to it, in the same way as a COM(x) port is actually a serial port set to COM(x). Usually an individual machine will have two LPT ports, although one can have up to three ports – LPT1, LPT2 and LPT3. Although they were designed primarily for printers, a wide variety of peripherals can be connected to these ports. Parallel port/interface refers to a standard 25-pin (D25) connector found on most (if not all) PCs, commonly used for connecting a printer. A standard parallel port transmits eight data bits at a time, as opposed to a serial port which transmits data one bit at a time. Due to the speed advantage (up to around 150 kbps a second) over the serial port, parallel ports are com-

monly used for printers and even small networks. ECP (Extended Capability Port) and EPP (Enhanced Parallel Port) are both implementations of the parallel standard; they both offer faster data transfer (up to 2 Mb/s) and are commonly supported in modern PCs. It is another simple interface standard designated as LPT port, configured for printer connectivity, on the rear side of the PC with 25-pin connector facility. This standard permits parallel communication of eight bit data (port address 0×378 for LPT1), five bit status (port address 0×379 for LPT1) and four bit control (port address $0 \times 37A$ for LPT1) between the PC and a system within a distance of 1 m, but faster compared to the serial interface. With this interface, the system has to be located near the PC.

USB port interface standard

Anyone familiar with computers would be aware of the problems that the USB is trying to solve – in the past, connecting devices to computers was a difficult task. For example,

- Printers connected to parallel printer ports, and most computers only came with a single parallel port. Zip drives, which need high-speed connection to the computer, would use the parallel port as well, often with limited success and not much speed.
- Modems used serial ports, but so did some printers and other devices like palm pilots and digital cameras. Most computers have at most two serial ports, and they are slow in most cases.
- Devices that needed faster connections came with their own cards, which had to fit in a card slot inside the computer case. Unfortunately, the number of card slots is limited and one had to install software for some of the cards.

The goal of the USB is to solve all these problems. The USB provides a single, standardized, easy-to-use way to connect up to 127 devices to a computer. Just about every peripheral made now comes in a USB version. A sample list of USB devices that one can buy today includes printers, scanners, mice, joysticks, flight yokes, digital cameras, webcams, scientific data acquisition devices, modems, speakers, telephones, video phones, storage devices such as zip drives and network connections. The USB interface is an ultrafast serial interface now dominating the field of automation and control system. It can support 127 USB devices at a time with unimaginable speed. USB 1.0 standard has the speed of 12 mbps; but USB 2.0 has the speed 40 times higher than USB 1.0, i.e. 480 mbps. On-line video cameras and video conferencing use this standard of interface due to high speed and the device connectivity. USB interface can be converted to any other standard like serial, parallel and GPIB (IEEE488 bus) bus with converter cables available in the market. Thus one can extend the number of standard ports making use of the USB hub connectivity.

Game port interface standard

This port permits connectivity of two joysticks for playing video games. It can also be used for interfacing simple systems.

Embedded design solution

The recent trend in hardware design is simple and supports all required tasks by means of a programmable single chip. The design is referred to as embedded design, meant for any specific requirement with the common interfacing standard to enable interconnectivity of the system to the host. As customer performance-expectations skyrocket, system designers must take care of updating processor and communication technologies for each new project. Yet, the problem with system revisions is the significant learning curve necessary for becoming proficient with a new architecture and associated development tools. It often takes months to uncover all of the nuances and fine details of a new tool set. To address these problems, most board- and silicon-level vendors offer low-cost evaluation or development kits that quickly demonstrate product performance and ease the transition to new hardware and software. The innovative programmable embedded hardware like field-programmable gate array (FPGA), application-specific integrated circuit (ASIC), programmable system on-chip (PSoC), digital signal processor (DSP) and several new microcontrollers, are examples of the embedded design solution. The contents of a design-support or evaluation kit vary depending on the featured product and the marketing approach the vendor adopts. Hardware ranges from a simple peripheral module that one can plug into a design to a stand-alone embedded sys-

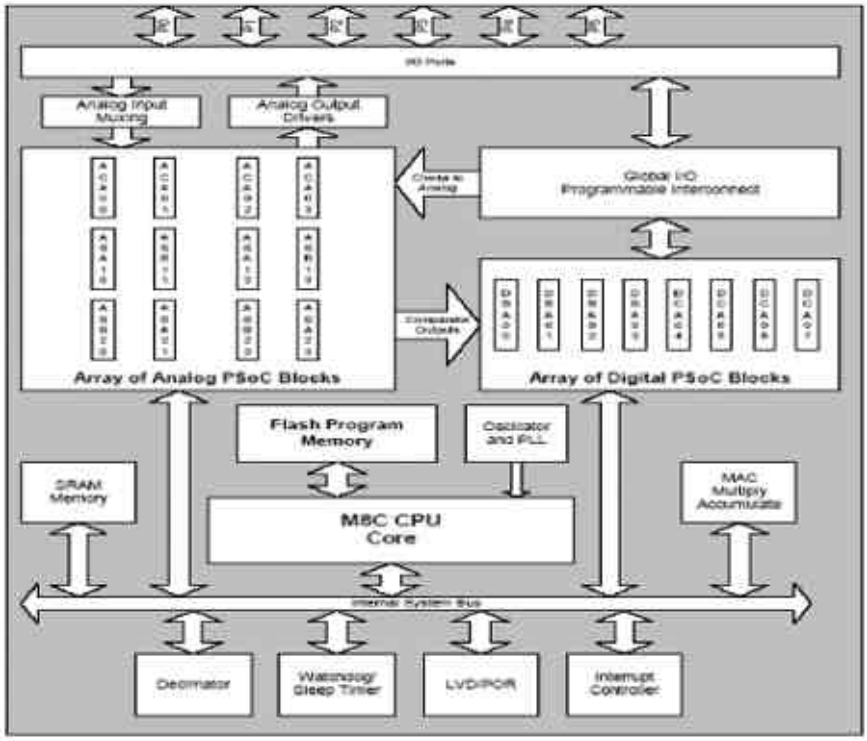


Figure 1. Internal block diagram of PSoC chip.

tem with processor, memory, peripherals, programmer, and bread boarding area to test one's application circuitry. Software offerings include drivers for unique silicon all the way up to a complete development environment for processor products. The main advantage in embedded design is that, before the target hardware becomes available, developers can use a software-processor simulator running on the host or a general-purpose evaluation board in place of the target prototype. At present, the increasing performance requirements need more complex interface designs with stand-alone components. Our goal is to make the complex designs easy for automation interface designs with novel, intelligent on-chip programmable components.

In this note Cypress Microsystem's PsoC-based application design with virtual instrument graphical language control program is described as an example.

Programmable system on-chip

PSoC (TM) mixed-signal arrays of cypress are programmable systems-on-chips (SOCs) that integrate a microcontroller and the analogue and digital components that typically surround it in an embedded system, as shown in Figure 1. A single PSoC device can integrate as many as 100 peripheral functions with a microcontroller of the customer saving design time, board space and power consumption. Easy-to-use development tools enable designers to select the precise peripheral functionality they desire, including that Analogue functions (amplifiers, ADCs, DACs, filters and comparators); Digital functions (timers, counters, PWMs, SPI and UARTs).

The analogue features of the PSoC family include rail-to-rail inputs, programmable gain amplifiers and up to 14-bit ADCs with exceptionally low noise, input leakage and voltage offset. PSoC devices include up to 32 kb of flash memory, 2 kb of SRAM, an 8×8 multiplier with 32-bit accumulator, power and sleep monitoring circuits and hardware 12C communications.

PSoC Designer(TM), the traditional software development environment for PSoC, is a fully-featured, GUI-based design tool suite that enables the user to configure design-in silicon with simple point and click options. With PSoC Designer, users can code the device in either 'C' or assembly language and debug the design using features such as event triggers and multiple breakpoints, while single-stepping through code in 'C' or assembly or a mix of the two languages.

Virtual instrument program

Virtual instruments (VI) are an application of general purpose digital PCs for the measurement and control of various parameters like temperature, flow, pressure, position, etc. VI programs mimic the control processes, which are in a remote area, on the PC screen. Experimentors/operators can visualize the on-going process control automation through the PC screen. VI programs provide an inexpensive, yet powerful platform for control and data acquisition of process variables. These programs are easy to implement with graphic languages (G-language) like LabVIEW, HP-vee, Bridge VIEW, Agilent VIEW, etc. The 'G' language implements data-flow technique. Usage of 'G' language VIs provides easy interfacing with PCs under the Windows environment[2].

VI programs can also be developed using any other text language like C, C^{++}, Pascal, Visual Basic, Visual C, etc. but require considerable effort to call hardware modules in the Windows environment. For example, building Dynamic Link Libraries (DLLs) for calling hardware port address in Windows environment is somewhat difficult. The 'G' language provides built-in function libraries for a variety of application requirements as graphic palettes, which in turn support the required DLLs for functions to run under Windows environment. Usually the 'G' language VI programs consist of two frames, viz. panel diagram and functional diagram. In

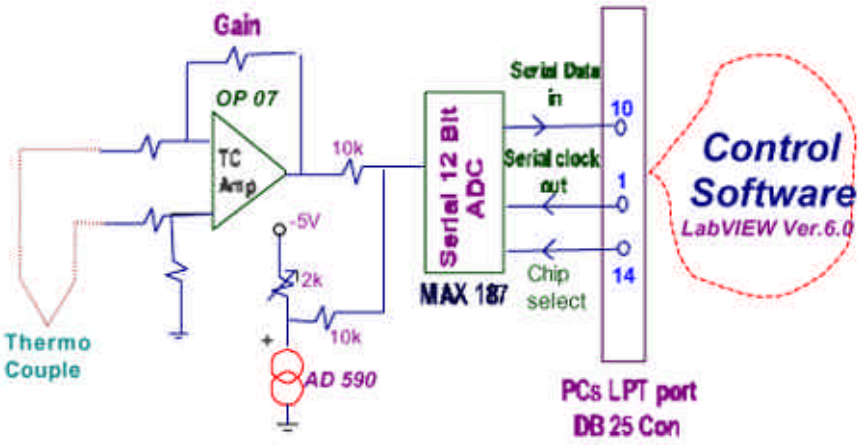


Figure 2. Simple design with discrete components.

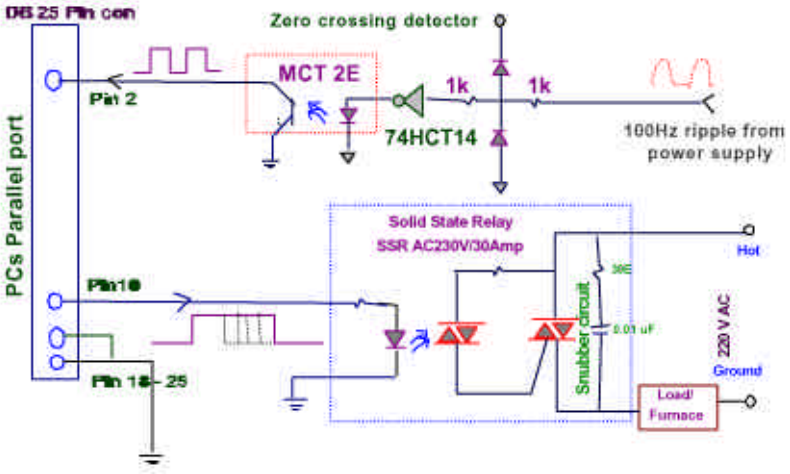


Figure 3. PWM power controller.

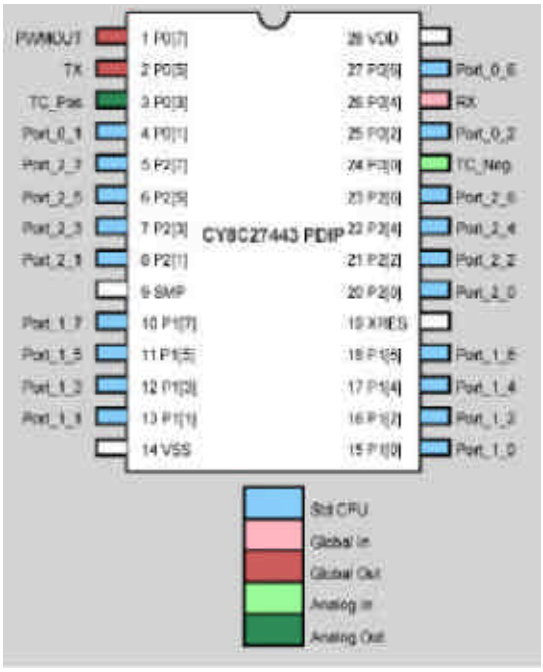
the panel diagram, programmers can assign various controls and indicators (i.e. input and output variables) according to their requirements and in the functional diagram, the designer can implement the required functions like mathematical operation, comparison, loop functions (**for** loop, **while** loop, **case** loop, **formula** node, etc.), decision-making, data conversion, file-copying sequence, input/output port addressing, implementing standard interfacing like parallel port, serial port, GPIB (IEEE488), USB, and also filters, curve-fitting functions, etc. For the purpose of networking Transmission Control Protocol (TCP), Internet Protocol (IP), data socket connectivity are also available as a function library in LabVIEW. Programmatically calling functions written in any other language is also possible from LabVIEW, by the usage of call library function node, code interface node (CIN), AcriveX functions. Similarly, any 'Matlab' routine can be called from LabVIEW using Matlab script node. The National Instruments' LabVIEW version 7.1 incorporates all the necessary functions as 'icons' in its package. Even beginners can write their own programs making use of this package within a reasonable time span in Windows environment.

Simple design with discrete components

A simple example shown here is a temperature controller[3], widely required in laboratory/industrial application.

Figures 2 and 3 show the temperature measurement and power control circuit diagram of a simple and novel design approach for measurement and control of temperature with minimal number of discrete components. This design uses the parallel port interface standard and control functions of the PC through VI software program written in *LabVIEW*, a graphical language. In this design (Figure 2), the thermocouple amplifier with suitable gain measures the temperature of the system under investigation. IC AD590 compensates room temperature changes and the final compensated temperature analogue information is converted to digital code by the serial 12 bit ADC MAX 187. The three-pin connectivity of the ADC to the PC through the parallel port communicates digitized data to the PC (Figure 2). With this acquired digital data, the VI

a



b

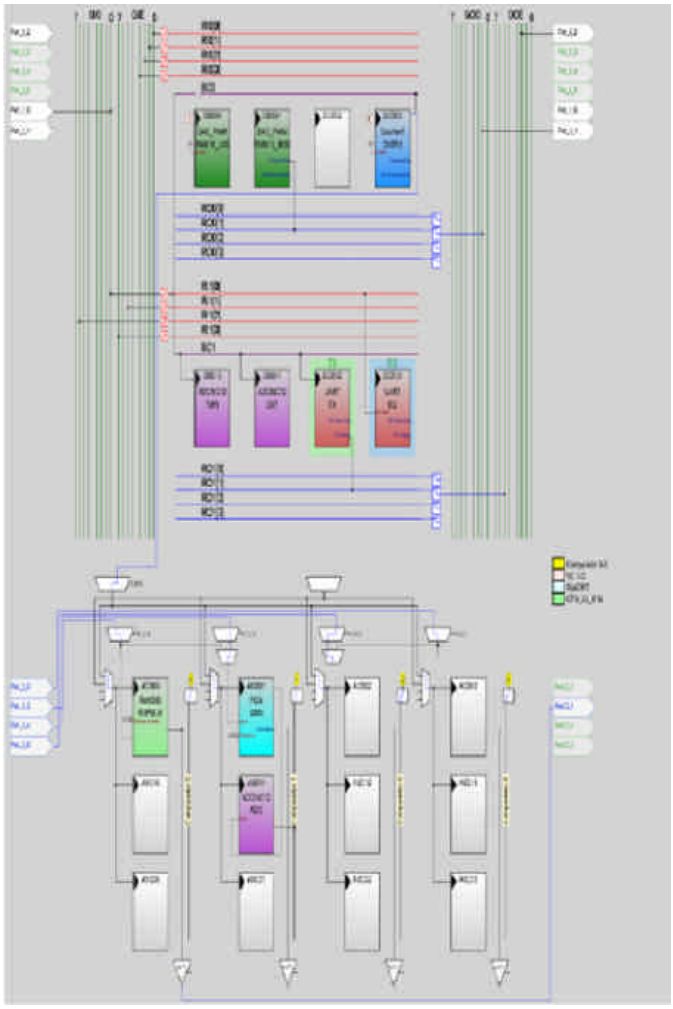


Figure 4. ***a***, PID Temperature controller in a single PSoC chip with pin diagram. ***b***, PID temperature controller PSoC block diagram.

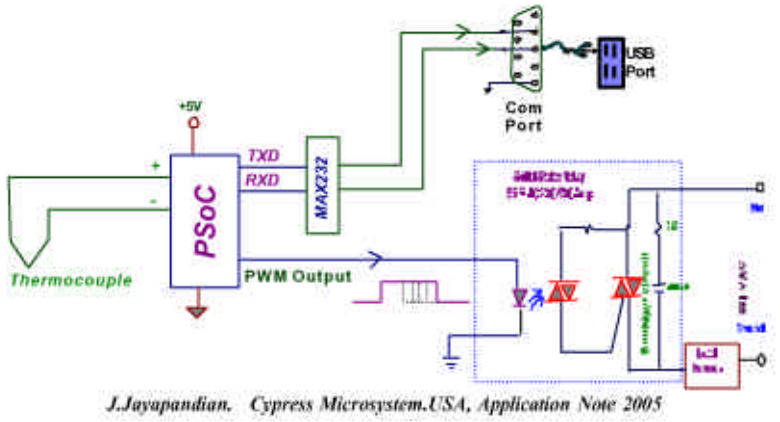


Figure 5. PID temperature controller in PSoC way.

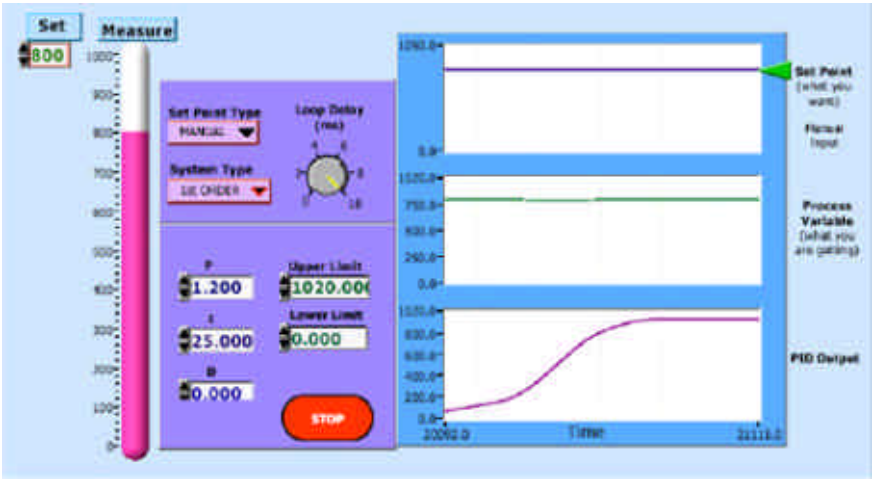


Figure 6. VI front panel diagram for PID temperature controller.

control program calculates control parameters like programmable (P), integral (I), differential (D), etc. and sets the required pulse width through a bit (pin 10) of parallel port and controls the AC cycle in a pulse width modulated (PWM) firing angle sequence for the power requirement via a solid state relay (SSR)[4,5] (see Figure 3). To enable proper firing sequence, the zero crossing in the AC cycle is detected by the opto-coupler circuit and sensed through another bit of a parallel port (pin 2). This simple design controlled by the VI program for control function provides an easy means of achieving temperature control with user-required programmable control functions like ON/OFF, PID, linear heating, on-sweep measurement, etc. A similar way of interfacing is possible with the serial port of the PC also. The embedded control example with PSoC describes the serial interfacing design for the temperature controller. The choice of type of interfacing is left to the designer/user.

Embedded design simplifies design automation

Temperature controller in PSoC way

The above design can be easily implemented in an embedded design as a programmable single chip using PSoC designer, as shown in Figure 4 *a* and *b*. Figure 4 *a* shows the pin configuration of a PSoC IC and Figure 4 *b* shows the internal analogue and digital blocks of temperature controller design in a single PSoC chip. Analogue modules programmable gain amplifier (PGA), a 12 bit ADC (analogue parts of ADC), a reference ground (analogue modules), a 16 bit PWM, UART block for serial communication with the PC and a 8 bit counter for serial baud rate select (digital blocks include ADC digital part) have been programmed in a single 28-pin PSoC chip, CY 827443. Figure 5 shows the miniaturized embedded temperature controller design in PSoC way. The serial communication with the PC enables the embedded system to interact with the PC through the VI control program written in LabVIEW[5].

VI program for PID controller

Figure 6 shows the front panel VI program panel diagram for PID temperature controller. The VI panel diagram is the same for simple design approach (shown in Figures 2 and 3) as well as for embedded control design (Figure 5). The only difference is the interfacing standard, i.e. in a simple approach the design uses the parallel port of the PC and in embedded control PSoC design uses the serial port

interface of the PC[6]. The function diagram of the VI program uses graphical icons required for parallel or serial ports according to the design consideration. The VI panel diagram shows user-friendly control over parameters required for proper control action of the temperature controller. Control parameters like proportional 'P', integral 'I', differential 'D', loop delay time constant can be tuned from the VI front panel itself. A sub VI can be built to automatically set the control parameters of the furnace under connection for auto-tuning facility. From the on-line graph one can identify the proper control function of the controller.

Conclusion

A simple interfacing technique using a PC for measurement and control of various physical quantities has been discussed. For one of the most common applications, namely, temperature measurement and control, a miniaturized embedded control design with implementation of VI graphical language control program has been discussed. This shows the simplest way by which any design requirement for laboratory as well as for industries can be exploited. Any required task can be implemented indigenously according to the user requirement with proper selection of sensors and its connectivity to the PC through embedded design and VI program technique. This approach improves the experimental technique with good accuracy and reliability. The existing trend in PC-based laboratory automation involves minimal hardware and maximal support of software components. With available PCs and intelligent programmable embedded hardware and software components, automation of experiments becomes simple, reliable and inexpensive.

1. Cypress Micro System, *Programmable System On-chip User Manual*. //www.cypress.com
2. National Instrument's *LabIVEW User Manual*. //www.ni.com
3. Jayapandian, J., *Electronic Design Magazine*, Penton, New Jersey, USA. ED Online ID # 1575, 4 March 2002.
4. Jayapandian, J., *Electronics Design Magazine*, Penton, New Jersey USA, ED Online ID #5687, 15 September 2003.
5. Jayapandian, J. *et al*., *J. Instrum. Soc. India*, 2003, **33**, 75–80.
6. Jayapandian, J., *Cypress Micro System Application note AN2287*, September 2005.

Received 22 December 2005; revised accepted 3 January 2006

J. Jayapandian is in the Materials Science Division, Indira Gandhi Centre for Atomic Research, Kalpakkam 603 102, India.
e-mail: jjpandian@gmail.com